Web Interface Development Environment (WIDE): Software Tool for Automatic Generation of Web Application Interfaces

Sohei Okamoto, University of Nevada, Reno, USA, okamoto@cse.unr.edu
Sergiu Dascalu, University of Nevada, Reno, USA, dascalus@cse.unr.edu
Dwight Egbert, University of Nevada, Reno, USA, egbert@cse.unr.edu

ABSTRACT

Using web applications has become a common solution to access and manipulate information remotely. Platform and device independency can be achieved most efficiently using matured web standards. By combining available standards and web technology, complex interactive web applications can be created. However, building even basic web applications is limited to expert software developers. Even though end-users have a desire to create one, the hurdle to learn and combine multiple technologies altogether is often too high for their needs and skills. In this paper we present WIDE, a new software application designed to assists end-users develop web applications visually and thus minimize their efforts.

KEYWORDS: End-user programming, visual programming, web design, web application development, prototyping, navigation charts

1. INTRODUCTION

The World Wide Web (WWW) standard has been widely spread, and the use of information technology has become common in everyday activities for many people to communicate remotely. By having rich user interaction on the web integrating server technologies, web applications can provide most of the functionality that desktop applications have been providing, with additional remote capability and platform/device independency. However, developing such applications requires expertise in web development frameworks and programming languages, which most people do not have or are not usually willing to learn in addition to their other work responsibilities.

The work presented in this paper is intended to provide a solution for end-users to build effective web applications intuitively and with minimum overhead.

There are several existing projects that are directed to achieve a similar goal, primarily flow visualization [3, 4, 5], sketching [7, 10], and end-user programming [13, 14]. There are also several attempts in using visual notations to specify web applications development [2, 6, 17].

Since the intentions and goals of each approach are slightly different, all of them have limitations regarding a solution for assisting end-users develop effective web applications.

In this paper, we present Web Interface Development Environment (WIDE), a new solution for building web applications. WIDE provides a development environment with an easy-to-use interface where users can design web applications by navigation charts, which significantly minimizes learning, development, and maintenance time.

Currently, WIDE is designed to develop form-based transaction-like web applications; nevertheless, by defining and adding component libraries to the system, the potential features of web applications that WIDE can generate are limitless.

The remaining of this paper is structured as follows: Section 2 provides an overview on background technology and related studies, as well as a brief analysis of related work. Section 3 describes the overall solution proposed mainly via the tool's software model expressed using the Unified Modeling Language (UML). Section 4 shows the implemented prototype solution in

action. A comparison with similar work is provided in Section 5. Planned future work and enhancements are described in Section 6, followed by the conclusions of the paper in Section 7.

2. BACKGROUND

Thanks to standardization and the wide spread of the Hyper Text Markup Language (HTML), digital materials are becoming popular means of collecting and publishing information, which leads to the maturity of visual editing tools – the so called WYSIWYG (What You See Is What You Get) editors, – tools that allow a novice user to create a web page without having knowledge of HTML. However, these tools are only sufficient for the creation of static web sites. Development of web applications, which are interactive programs on the web to provide services and solutions to problems, currently requires learning several different languages and technologies. A study of non-programmer web application development [12] suggests that the limitation for end-users to create a web application is not due to lack of interest, but to difficulties in the development process and environment.

In studies of end-user programming [8, 9] several cognitive factors that cause challenges and learning barriers are identified. The authors suggest that end-users have significant difficulties in learning and integrating multiple languages, in addition to difficulties in understanding program's behavior by analyzing the source code.

Several approaches have been suggested to minimize the challenges of application developments. One of the *visual programming* approaches is *iconic programming*, where icons are used to create flowcharts that enable users to quickly develop and experiment with new algorithms visually [4, 5]. Another solution in visual programming is *sketching*, where users can draw diagrams using a pen interface to facilitate fast prototyping and interface designing in web development [7, 10]. The use of *modeling languages* specialized for web applications has also been suggested to define concrete methods to develop large and complex web applications from requirements specification and design documents [2, 6, 17]. For example, **HopiXForms** [3] proposes the use of form and Model-Viewer-Controller (MVC) to define page presentation and program control flow. Furthermore, *end-user programming* addresses common human factors in usability and promotes combining intuitive interfaces and concept models, for example as applied in **Click** [13] and brain-image-oriented programming (**BioPro**) [14], where intuitive interfaces and wizards are provided to create pages with database manipulation. Both **Click** and **BioPro** allow users to visualize relationships between components in the application as well as to preview and test the system at any time, which helps users in understanding and debugging the application.

These approaches have advantages that are unique and effective in certain situations; however, all of them are not quite suitable in end-user web application development. Iconic programming solutions, including its leading system **RAPTOR** [4], are intended as rather algebraic code visualization tools than application development tools, while a sketching-based system such as **DENIM** [7, 10] is suitable for designing static web sites but not for developing web applications. Using a modeling language requires users to learn multiple non-intuitive frameworks, architectures, and languages. End-user oriented and visual programming solutions such as **Click** and **BioPro** are the most suitable systems in terms of creating web applications, but they are limited to basic solutions and cannot be used for prototyping and designing larger applications.

3. WIDE: Web Interface Development Environment

One of the main characteristics of the WIDE system is the capability of automatically generating web applications using navigation chart definitions. Navigation charts are one of the most intuitive methods to prototype applications, especially for non-expert users. Such users can construct a program's flow using their conceptual models, not necessarily knowing the exact details from the very beginning. After the framework of the application flow is defined, they can add details to each state of the navigation chart iteratively. To facilitate this process, the WIDE

system provides a mechanism of layered viewing. Users can view the navigation chart in high-level or abstract mode to overview the whole structure of the application they are building, as well as zoom into each state incrementally to see details of the state's flow. This helps users follow their own mental model of the application flow, while at the same time they can focus on implementing specific layouts and behaviors of a web page.

3.1 Main Functions

The main functions of the WIDE environment are grouped in three categories. First, the graphical user interface components provided in the environment process the user's actions to create a navigation chart. Users are allowed to change states and edit navigation charts with the mouse in the interactive drawing interface of WIDE. Second, WIDE has a server module to generate and update navigation charts and web applications as well as to send and run the application upon viewers' request. Third, the WIDE system has a communication manager to send requests to and receive responses from a server to operate file manipulation and data management, such that a navigation chart and web application can be updated and presented instantaneously upon user request.

3.2 Requirements and Use Case Modeling

Due to space limitation, for details of functional requirements, non-functional requirements, use case diagram, use cases, class diagram, and requirements traceability matrix the reader is referred to [15].

3.3 High Level Architecture

The WIDE system is composed of two modules, the client and server, each of them further divided into sub-modules, as shown in Figure 1.

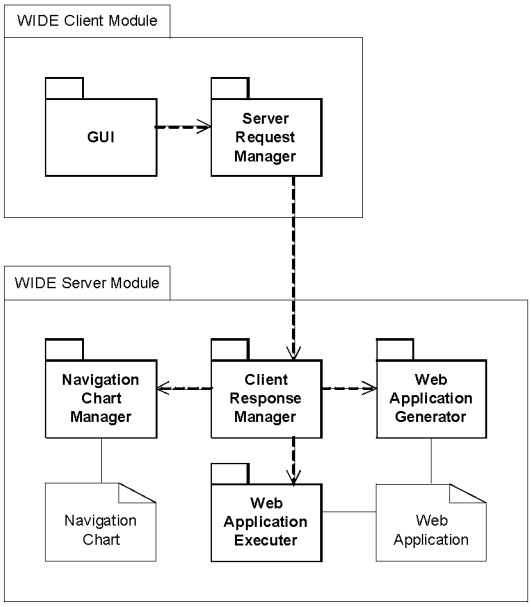

Figure 1. Client and Server Main Modules.

The client module consists of the GUI module and the server request manager, whereas the GUI module provides the interactive interface to users and relies on the server request manager to update a navigation chart or a web application. The server request manager sends requests to the client response manager in the server module while the client response manager utilizes a

navigation chart manager to process navigation chart requests, a web application generator to automatically create web pages, and an executer module to process web application requests. The client module was developed in Flash and Actionscript, whereas the server module was developed in PHP and its Ming package to dynamically generate data.

3.4 AI Module

An artificial intelligent (AI) module is currently being designed to be integrated in the WIDE system to enhance the automation of the development process. Reusing common set of components or repetitive actions are often observed in web development. Users perform this behavior extensively by integrating existing components or libraries. However, defining the user's own library or integrating external frameworks and libraries requires expertise in application development. To facilitate the process of reuse, the AI module is proposed to catalogue common tasks by learning users' behaviors and to suggest appropriate tasks when users perform similar sequences of actions. The system will record the actions performed by a user in designing the application's presentation and behaviors and after several session will analyze them to obtain a generalized set of actions commonly performed by the user. For example, a particular user may tend to create pages with title, text, and buttons placed at certain positions and may like to add the same specific transitional actions to each commonly used button. These repeated actions will be catalogued as tasks with additional information attached, such as previous actions or frequency of task repetition. By comparing previously catalogued sequences with new actions, the system will be able to provide suggestions for suitable set of actions to be performed, which will significantly reduce repetitive development procedures. The AI module can be considered an instrument for automatic generation of macros that defines tasks by learning the user's behavior and by suggesting performing appropriate tasks when the user is attempting similar ones.

4. PROTOTYPE

The main interface of the WIDE system, shown in Figure 2, consists of the menu bar and the tool bar at the top, the drawing area in the middle, and a slider control to zoom the drawing area on the left.

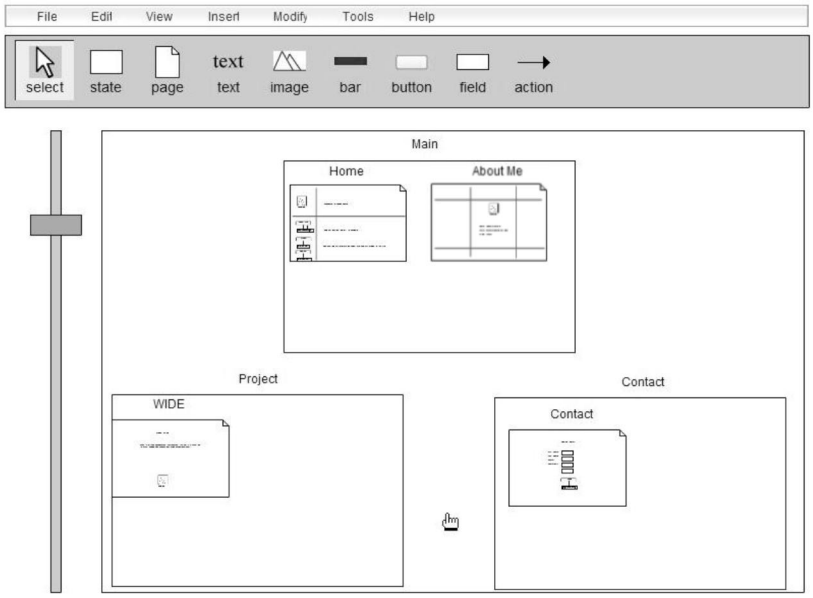

Figure 2. WIDE Main Window with Navigation Chart.

Using the tool bar, users can insert navigation chart elements into the drawing area. The figure shows how the navigation chart looks after some designing. Note that the slider control on the left indicates that the drawing is in the high-level (or abstract) mode of operation, where users can grasp the overall structure of the application.

The next figure shows the zoom level changed in order to view more details of the states. One of the states is shown in Figure 3, where the abstract page layout and the components are visible inside, with the action component set available to specify the transitions.

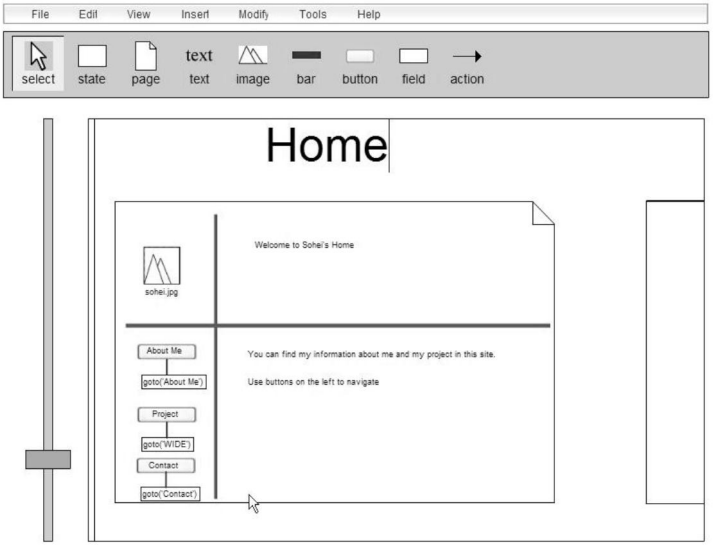

Figure 3. WIDE Main Window with Detailed Layout.

When the user chooses to publish the pages, a web site is generated with layout specified in each page and transitions described in action components. The system generates a SWF format file which is integrated in an HTML page. One of the generated pages is shown in Figure 4, where buttons in the page trigger transitions to the other pages.

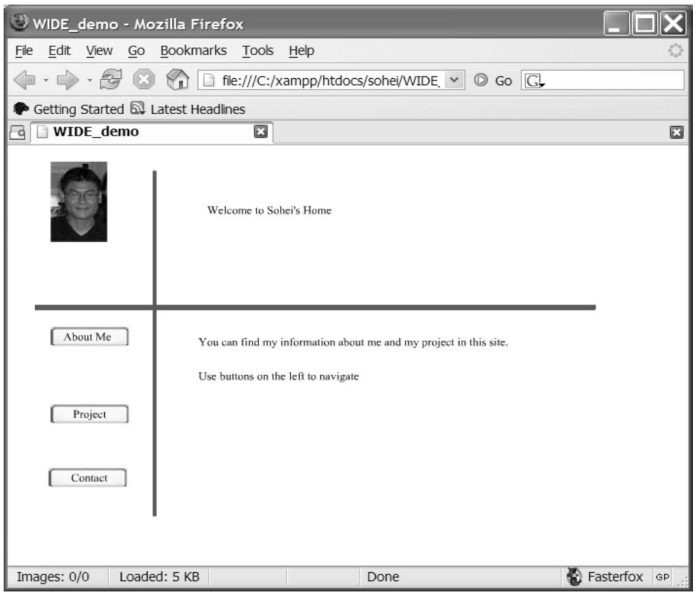

Figure 4: Generated Home Page.

5. COMPARISON WITH RELATED WORK

The comparison is made with work that is specifically designed for end-users to generate web sites or web applications. The programs taken into consideration for this comparison are **DENIM** [7, 10], **Click** [13], and **BioPro** [14].

First, **DENIM** and **BioPro** are based on JSP, which limits their capabilities of application deployment. Moreover, the end-users need to have some knowledge of Java when they generate the web sites.

In addition, **DENIM** is designed to prototype rather static web sites, and not dynamic web sites, which is a necessary feature for newer web applications. **Click** can generate dynamic web sites with database integration, but it is limited to build the sites page by page, which is tedious when building large web applications. Furthermore, the actions that users can employ are limited to predefined actions provided by the program. **BioPro** is intended to create larger and more complex web applications with database integration, but its interface is only effective when the applications do not incorporate complex interactions.

Thus, the existing similar works are not fully suitable for end-users to build larger and more flexible web applications, where WIDE overcomes more of the development-related difficulties. Table 1 provides further comparison in terms of essential features for generating dynamic websites and web applications: navigational flow, zoom facility, and widget-based actions.

Approach	Flow	Zoom	Action	Dependency
WIDE	Yes	Yes	Yes	Flash Player / PHP
Click	No	No	Yes	Javascript / PHP
BioPro	Yes	No	Yes	JSP
DENIM	No	Yes	No	JSP

Table 1. Comparison with Similar Work.

6. FUTURE WORK

The WIDE prototype provides an easy-to-use, flexible interface for building web applications based on navigation charts. As this is an early version of the WIDE development environment, web applications that users can build with it are still limited. Thus, a number of enhancements are planned for the near future, as follows.

First, an important enhancement will be to add a scripting feature which should be easy enough for end-users to write with.

Second, another useful enhancement will be to add database management features, such as creating and using tables in web applications. Intuitive data visualization and management methods are essential for building effective web applications.

Third, yet another significant enhancement will be to add a component library for common application component-based development.

Lastly, several formatting features such as advanced text formatting and layout styling will also be needed in the future.

7. CONCLUSIONS

The main goal of this paper was to propose a new tool for end-user development of dynamic-contents web applications. To allow end-users to build such applications an intuitive and effective process as well as an easy-to-use web interface development environment were required. The WIDE environment, presented in this paper, allows users to create navigation charts using their own conceptual models and automatically generate web applications without performing complicated tasks. By using WIDE, users are not required to learn multiple architectures and

technologies, whereas this is not the case in most of the other existing similar frameworks. In addition, WIDE provides a zoomable interface, such that users can easily and intuitively change the levels of detail, from abstract to detailed views. By using this feature, users can keep their conceptual model clear and unaltered, while iteratively adding complexity to the applications they develop. This feature is also beneficial to expert users in their demanding task of constructing large, complex web applications.

As far as we know, there is no other program available that allows a broad range of end-users to build dynamic web applications via an intuitive and easy-to-use specialized development environment. The WIDE environment offers a friendly interface and a cost-effective, streamlined and practical solution for end-user web application development.

8. REFERENCES

[1] Arlow, J. and Neustadt, I., *UML and the Unified Process: Practical Object-Oriented Analysis and Design*, Addison-Wesley, 2002.

[2] Bochicchio, M., and Fiore, N., "WARP: Web Application Rapid Prototyping," Proc. of the 2004 ACM Symposium on Applied Computing (SAC '04), 2004, pp. 1670-1676.

[3] Cardone, R., Soroker, D., and Tiwari, A., "Using XForms to Simplify Web Programming," Proc. of the 14th Intl. Conf. on World Wide Web (WWW '05), 2005, pp. 215-224.

[4] Carlisle, M.C., Wilson, T., Humphries, J.W., and Hadfield, S.M., "RAPTOR: A Visual Programming Environment for Teaching Algorithmic Problem Solving," Proc. of the 36th SIGCSE Technical Symposium on Comp. Sc. Education (SIGCSE '05), 2005, pp. 176-180.

[5] Chen, S. and Morris, S., "Iconic Programming for Flowcharts, Java, Turing, etc," Proc. of the 10th Annual SIGCSE Conf. on Innovation and Technology in Computer Science Education (ITiCSE '05), 2005, pp. 104-107.

[6] de Andrade, A.R., Munson, E.V., and Pimentel, M.G., "A Document-based Approach to the Generation of Web Applications," Proc. of the 2004 ACM Symposium on Document Engineering (DocEng '04), 2004, pp. 45-47.

[7] DUB. University of Washington, DENIM, http://dub.washington.edu/projects/denim/

[8] Ko, A., Myers, B.A., and Aung, H.H., "Six Learning Barriers in End-User Programming Systems," Proc. of the 2004 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC '04), 2004, pp. 199 – 206.

[9] Ko, A.J. and Myers, B.A., "Human Factors Affecting Dependability in End-User Programming," The International Conference on Software Engineering, Proc. of the First Workshop on End-User Software Engineering (WEUSE I), 2005, pp. 62-65.

[10] Lin, J., Thomsen, M., and Landay, J.A., "A Visual Language for Sketching Large and Complex Interactive Designs," Proc. of the SIGCHI Conf. on Human Factors in Computing Systems (CHI '02), 2002, pp. 307-314.

[11] The PHP Group, PHP: PHP Hypertext Processor, http://www.php.net/

[12] Rode, J., "Nonprogrammer Web Application Development," Proc. of the SIGCHI Conf. on Human Factors in Computing Systems (CHI-2004), 2004, pp. 1055-1056.

[13] Rode, J., Bhardwaj Y., Perez-Quinones, M.A., Rosson, M.B., and Howarth, J., "As Easy as 'Click': End-User Web Engineering," Proc. of 5th Intl. Conf. on Web Engineering (ICWE 2005), Springer-Verlag, LNCS 3579, 2005. pp. 478-488.

[14] Shimomura, T., "Visual design and programming for Web applications," *Journal of Visual Languages & Computing*, Vol. 16, Issue 3, 2005, pp. 213 - 230.

[15] Sohei Okamoto's thesis, http://www.cse.unr.edu/~dascalus/theses/soheiokamoto2005.pdf

[16] OMG's Unified Modeling Language, UML Resource Page, http://www.omg.org/

[17] WebML, The Web Modeling Language, http://www.webml.org/